# An Ontology-based Intelligent System for Medical Diagnosis

**Naoufel Khayati, Wided Lajouad-Chaari**

*Abstract*— **We present a distributed collaborative and intelligent system assisting physicians in diagnosis when processing medical images. This is a Web-based solution since the different participants and resources are on various sites. Communication between these different components is ensured by mobile agents. Furthermore, in order to share a common understanding of our application domains and to solve problems related to the semantic of the queries, we have equipped our distributed system with ontologies. Our system is collaborative because the participants (physicians, radiologists, knowledge-bases designers, program developers for medical image processing, etc.) can work collaboratively to enhance the quality of programs and then the quality of the diagnosis results. It is intelligent since it is a knowledge-based system including, but not only, a knowledge base, an inference engine said supervision engine and ontologies. The current work deals with the osteoporosis detection in bone radiographies. We rely on program supervision techniques that aim to automatically plan and control complex software usage. Our main contribution is to allow physicians, who are not experts in computing, to benefit from technological advances made by experts in image processing, and then to efficiently use various osteoporosis detection programs in a distributed environment.**

*Index Terms*— **Program Supervision, Distributed Program Supervision Systems, Mobile Agents, Knowledge Model, Ontologies, Medical Imaging, Osteoporosis Detection.**

## I. Introduction

Medicine is considered as one of the large application fields of the image analysis and its processing. For example, image analysis is widely needed in the imagery by magnetic resonance, in the radiology to assist physicians in their diagnosis, and recently in the telemedicine. This analysis is considered by specialists in image processing in order to offer more effective programs and more efficient approaches. We experiment our work in the osteoporosis detection. The most challenging task is to characterize bone micro rchitecture by parameters that can be automatically estimated from radiographies and that can accurately detect and quantify alterations of bones. For this, an original approach using morphological tools to extract characteristic features of trabecular bone images has been developed [11]. To make such an approach for medical diagnosis, we determined an "image protocol" adapted to bone types (e.g. femur, wrist, vertebrae) and patient types (e.g. male or female, adult or child) for various image resolutions [4][6]. Setting such an image protocol consists in planning a sequence of programs and tuning their input values (e.g., threshold values, filter size, etc.). This constitutes a tedious, time consuming task which requires both clinicians and image processing experts to collaborate. Our solution was to provide an interactive tool which relies on artificial intelligence techniques to build image protocols in different situations. Moreover, we wish to make this system accessible to bone radiologists, a geographically scattered community. In previous work, we validated our technological and architectural choices and the morphological analysis [5][6]. In this paper, we investigate further, the use of mobile agents, and our knowledge models choices in terms of ontologies. Hence, this paper summarizes all that we have done in order to offer a *distributed intelligent assistant* for osteoporosis detection, relying on mobile agent technology.

**Naoufel Khayati**, University Assistant, University of Sousse - National School of Engineering of Sousse / COSMOS Laboratory - Univesrity of Manouba, Tunisia; +216 97 830 674.
**Wided Lejouad-Chaari**, Assistant Professor, COSMOS Laboratory, University of Manouba, Tunisia.

## II. Program Supervision

Several program libraries have been developed by specialists in various domains with an aim of automating the image processing. But, the user of these libraries of image processing and medical imaging does not have competences in data and image processing allowing him to use them in an effective way. Moreover, users (physicians) must focus themselves on the interpretation of the results and not on the way in which these programs are carried out and scheduled. Thus, approaches of Artificial Intelligence were proposed in order to assist a non-specialist in data processing for correct use of these programs in its field. These approaches are known as "Program Supervision" [9] which consists of the automation of management and the use of preexistent programs. These programs are considered as "black boxes" and their application domain or their programming language is not relevant. The goal is not to optimize the programs themselves, but to assist program *usage* [9].

To carry out a supervision task, a subset of programs is chosen, scheduled, and applied to a specific problem. This selection and this scheduling in various configurations are ensured by a supervision system, which, thanks to the reasoning of its engine and the knowledge contained in its base can free the user (the physician) to make this management manually. This enables a physician to run programs, to check the consistency of some image analysis methods, to compare algorithms, to evaluate results, to reconsider some parameters and to readjust them. Program supervision may be applied to different domains related to image, signal processing, or scientific computing like Astronomical Imaging (e.g. automatic galaxy classification [10]), Vehicle Driving Assistance (e.g. road obstacle detection [7]), and Medical Imaging (e.g. chemotherapy follow-up based on Factorial Analysis of Medical Image

Sequences [1][8], and the segmentation of 3D MRI images of the brain [1][8], osteoporosis detection in bone radiographies [4][5]).

## III. Architecture of the Distributed and Intelligent System

In our previous applications we used a centralized supervision system in which data, programs and the knowledge-based system (KBS) have the same location. However, applying program supervision to real applications as osteoporosis detection, require distributing it. Indeed, both data (images from different hospitals) and programs (developed by different teams) come from various places. In most cases it would be fairly inefficient to move data and executable code to the same place as the KBS, and in some case it is even not possible (e.g., programs may execute only on a specific hardware).

Therefore, we have developed a distributed version of the assistant, based on mobile agents, where clinicians from different countries may safely use the system and work collaboratively (run programs of other teams, check consistency of image analysis methods and evaluate results); where medical imaging experts may manage different versions of their programs; and where knowledge engineers may capitalize knowledge and adapt it to program changes. The distributed system, whose architecture is proposed in [2][3] and given by figure 1, is a triple (S, A, KM) defined by:

- S, a set of three types of servers
    - A session server playing the role of an interface allowing end-users to access the supervision services and to communicate with the other components of the distributed system.
    - A set of resource servers hosting programs, knowledge and supervision engine of the centralized system: a supervision server hosting the supervision engine, some program servers, some knowledge servers and some data servers.
    - And possibly a set of execution servers, on which programs are executed. For example, when dealing with MatLab programs, their execution requires at least, the presence of the MatLab tool on one of the servers.
- A, a set of agents who are responsible for updating the previous components and for performing requests. This multi-agent system combines stationary and mobile agents.
- KM, a set of knowledge models in the form of metadata and ontologies used to locate resources in order to define the mobile agents itinerary, to define access permissions and to analyze the user request so that it is properly treated.

## IV. Agent Model

When distributing the supervision system, mobile agents are used to implement the communications between its components. Metadata help localize the various resources (programs, data, execution servers, etc.).

The architecture of the multi-agent system [2][3] as presented on figure 2 consists of several types of servers. First, one *Program Supervision Server* runs the supervision engine named PEGASE; then possibly several *Execution Servers* enable to execute the planned programs; *Resource Servers* contain remote resources needed by executions (e.g., data files, scripts); finally, the end-user interacts directly with a *Session Server*.

The agents are classified according to their roles into three categories: *Interface agents (IA), Processing agents (PA)* and *Communication agents (CA).*

### A. Supervisor Agents (IA)

They are called also Session Managers; they are stationary and are coupled with the access web server. Each one manages the whole user session and the whole process of solving the supervision query in its charge. This means that there is one Supervisor agent by query. Such an agent:

- Reads the metadata to identify and localize the involved resources in the resolution of the query;
- Creates the other stationary agents for interfacing with the supervision engine (Engine agent) and the different execution servers (Execution agents);
- Determines the number of needed Solver agents and creating them. This computation is done according to some information given by the Supervision Engine: the *Ndep* dependency sets and the *Npar* parallel operators by set;

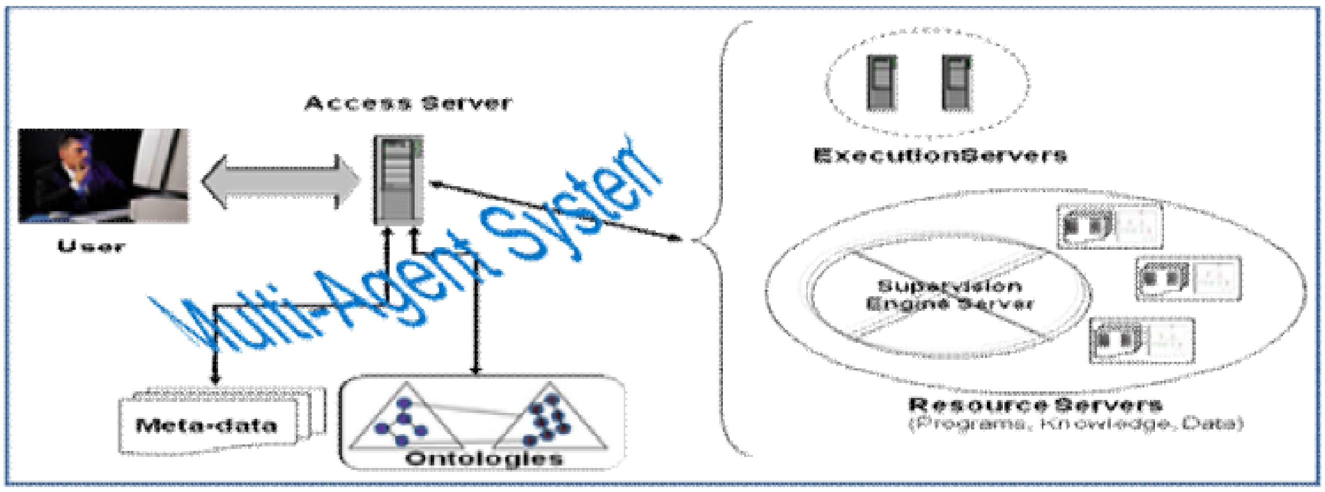


**Fig. 1.Distributed Assistant Architecture.**

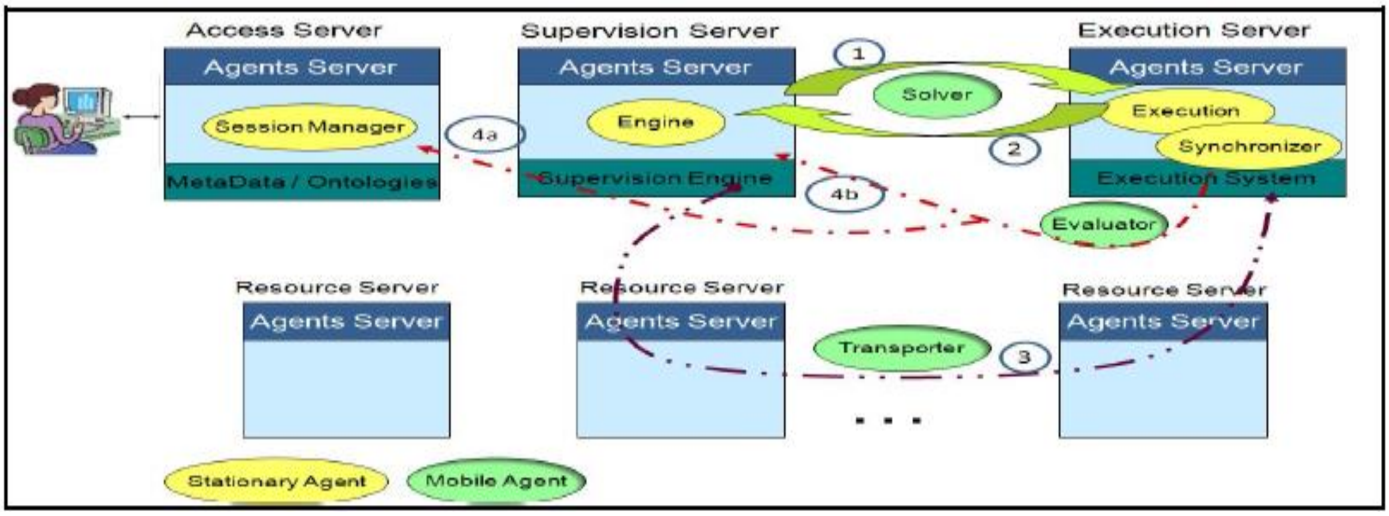


**Fig. 2.Architecture of the Multi-Agent System and Scenario of Use.**

For the last point, in a YAKL code [12] (Yet Another Knowledge Language, used by PEGASE), we may note parallel tasks in the Body section of a composite operator. We may also note dependency links between the sub-operators of a composite one, in its Flow section. These concepts are necessary when determining the number of solver agents. The dependencies will allow building some dependency sets that will allow finding the right number of solver agents to create. We define a dependency set as a set of operators having at least one common parameter.

### A. *Engine Agents (IA)*

There is an Engine Agent by query; it represents an "instance" of the supervision engine. The role of such an agent is to interface with the supervision engine by submitting to it, a query to process and getting from, a plan of programs to execute. Furthermore, it communicates the parallel treatments and the dependency sets to the Supervisor agent.

### B. *Solver Agents (PA)*

They are mobile and have to launch the execution of the remote programs already planned by the supervision engine. For example, while being under the control of the Supervisor agent, the Solver agent migrates to the supervision server (path 1 on figure 2) looking for the next instruction (the program and its call syntax). Then it seeks the necessary resources for this instruction from the Transporter agents, migrates to the corresponding execution server (path 2 on figure 2), and then waits until all the resources are available (information received from a synchronizer agent). At this stage, it starts executing the instruction, and finally, it sends the results to the Supervisor.

In general, the number of solver agents is determined according to these rules:

- For a dependency set of *N* elements, if it contains *Npar* parallel elements ($Npar \leq N$), then the maximum number of solver agents will be equal to *Npar* (one agent per parallel task, then one of them will continue with the other tasks of the same set). Otherwise it will be equal to 1.
- For *Ndep* dependency sets the maximum number of solver agents will be equal to $\sum_{i=1}^{Ndep} Npar\_i$. If no set has parallel tasks, such number shall be equal to *Ndep*.

### C. *Evaluator Agents (PA)*

They are created by Solver agents on the program sites or on the execution sites. They are created only if they are needed, i.e. if some programs require the evaluation of their results. An agent of this class stores the result to evaluate in its context and then must go to another server to perform its task. For its migration, the destination depends on the assessment type. Thus, if the evaluation is automatic, i.e. made by the supervision engine based on the knowledge base rules, it must migrate to the supervision server (path 4b on figure 2).
Otherwise, if the assessment is interactive, i.e. it requires the user intervention; it migrates to the access web server (path 4a on figure 2). In the case of an interactive assessment, the user response will be sent to the engine agent so it can decide the next step (continue with a new program or re-run the same program with repaired values for its parameters).

### D. *Transporter Agents (CA)*

They are created by the solver and are responsible for searching the necessary knowledge for the supervision engine in order to select the programs and their sequencing. In addition, they search the necessary resources to execute the current instruction, and transport them to an execution server (path 3 on figure 2). Since an instruction may need resources located on different nodes, there may be, simultaneously, several active Transporter agents.

### E. *Synchronizer Agents (CA)*

They are created by the Solver agent to synchronize the operations of the resources transport performed by the Transporter agents. Indeed, since many Transporters may be active simultaneously, they must register with the associated Synchronizer. When they are all registered, this agent starts them and waits until they do their jobs. Finally, when all the needed resources are available, it indicates to the Solver that it can continue its work.

## V. KNOWLEDGE MODEL

### A. *Knowledg Base Content*

For our tests, the program supervision system for osteoporosis detection, has supervised programs written in MatLab. The knowledge base (KB) is developed using the dedicated knowledge representation language, named YAKL [12], which uses both object-based and rule-oriented descriptions. This language allows experts to define domain types, objects or operators, and different rules to be used during reasoning.

For example, it defines the scheduling of the programs and describes the inputs and outputs of each one with its arguments and syntax. These descriptions are provided by the expert in medical image processing and are transparent to clinicians who have only to provide the digital radiographic images to be processed. A KB can be presented by a tree where the root is the principal composite operator, the intermediate nodes are the other composite operators and the leaves are the primitive operators. Figure 3 shows a simplified tree of our KB about the osteoporosis detection programs. The entry point of this tree is represented by the root operator *OsteoMorph*. This composite operator is decomposed into a sequence of primitive operators (oval forms) and composite ones (rectangular forms). The tree corresponding to our KB has six abstraction levels with 31 operators (11 composite operators and 20 primitive ones). Three of the eleven composite operators present a choice between operators, the other eight ones including one iterative operator, present sequences of operators. Among the primitive operators, there are:

- Three optional operators, i.e. their planning depends on the corresponding optionality criteria;
- Seven operators belonging to branches that require a choice, i.e. their planning depends on some choice criteria.
- Ten that are planned in all cases.

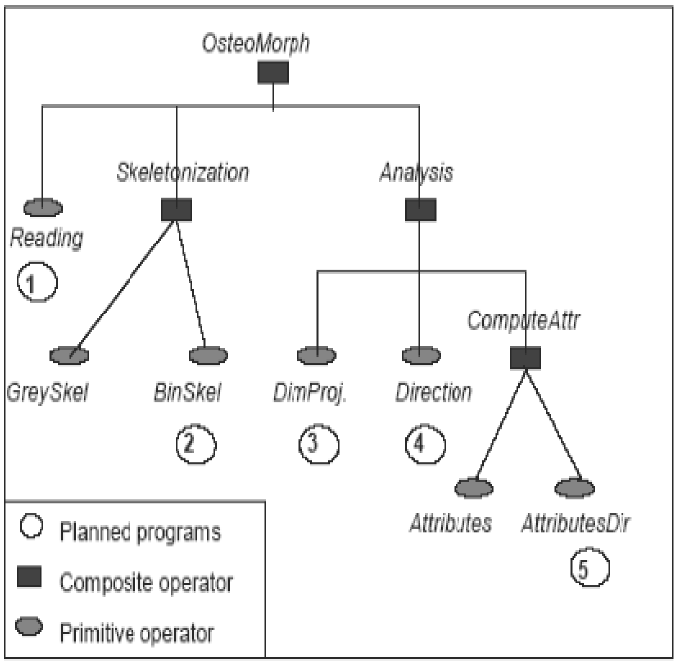


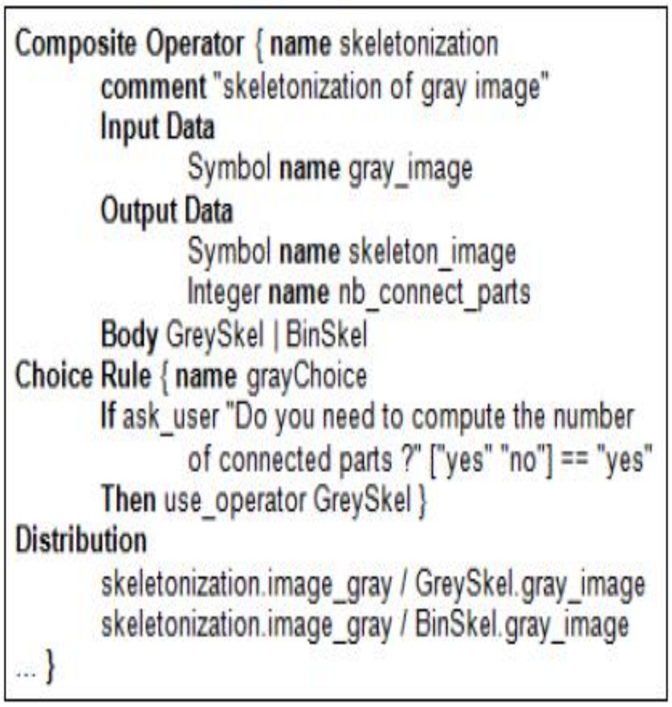
**Fig. 3.The OsteoMorph operator decomposition.**

Adding to this, there are about ten decision criteria. For instance, figure 4 gives an example from the osteoporosis knowledge base: a composite operator that describes an alternative decomposition (denoted by a |) into two suboperators: grey-level or binary skeletonization. A choice rule has been given by the expert to decide which sub-operator should be selected depending on the situation (in this case the choice is left to the end-user).

The *Distribution* part displays information about data transfer between the parent operator and its sub-operators. In other examples, when a composite operator is decomposed into a sequence of operators, the *Flow* part is added and will contain data transfer between the different sub-operators. The role of rules is essential to the engine strategy at different points during the reasoning process: for instance, to choose between several alternatives (choice rules, as shown above), to adapt program execution, to assess result quality, and to repair a badly assessed execution. These decision points are the key to establishing and adapting the "image protocols".

### B. *Ontological Architecture*

In order to share a common understanding of our application domains and to solve problems related to the semantic of the supervision queries, we have equipped our distributed supervision system with ontologies. The global ontological architecture [2] (figure 5) of our distributed system consists of three interoperable ontologies: an ontology for the supervision
domain and its knowledge, a second for the elements involved in its distribution (distribution ontology) and a third for the application domain (medical imaging and osteoporosis detection). For their integration into the system, our ontologies undergo the application of a reasoner which offers, in case of inconsistency, possible repair operations. Thus, we obtain semantic files reflecting them (Step 1 on figure 6).



```
Composite Operator { name skeletonization
        comment "skeletonization of gray image"
        Input Data
                Symbol name gray_image
        Output Data
                Symbol name skeleton_image
                Integer name nb_connect_parts
        Body GreySkel | BinSkel
Choice Rule { name grayChoice
        If ask_user "Do you need to compute the number
                of connected parts ?" ["yes" "no"] == "yes"
        Then use_operator GreySkel }
Distribution
        skeletonization.image_gray / GreySkel.gray_image
        skeletonization.image_gray / BinSkel.gray_image
... }
```

**Fig. 4. A composite operator in a YAKL code.**

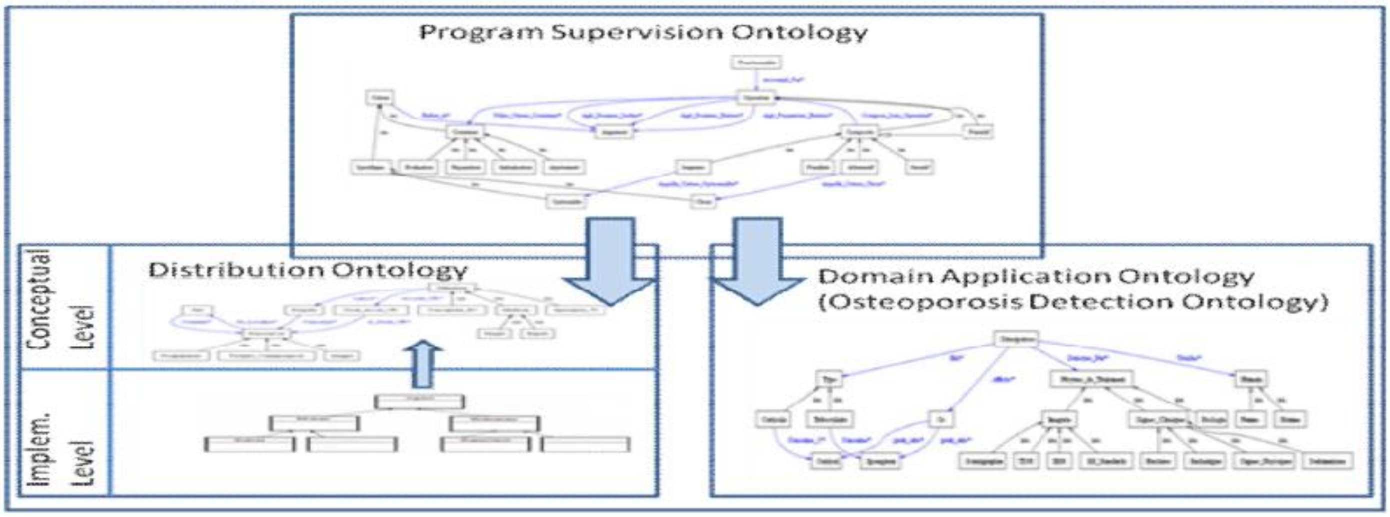

Program Supervision Ontology
Distribution Ontology
Domain Application Ontology
(Osteoporosis Detection Ontology)
Conceptual Level
Implem. Level

**Fig. 5. Ontological Architecture.**

## A. *Asking the Ontologies*

The query language defines the syntax and the semantics required to express queries and the possible forms of the results. We expressed interrogative SELECT queries type, which extract a sub-graph corresponding to a set of resources satisfying the conditions specified in the WHERE clause. Indeed, the user expresses his query in textual form, for example, "check the status of the bone".

This sentence will be split into words; neutral words will be eliminated (the, of, etc.). Then, the application domain ontology will be queried. This gives that the word "check status bone" is subsumed by the "Osteoporosis" concept. For testing, we added white concepts such as "diabetes", "Hepatitis", etc. to ensure that the query asked to the ontology receives the good answer. Then, the supervision ontology has to fetch the functionality which is responsible for the osteoporosis detection and subsequently the appropriate composite operator (Step 2 on figure 6). Once determined, this functionality will be communicated to the supervision engine (Step 3 on figure 6) so it can decide the "good" knowledge files (Step 4 on figure 6) and thereafter, the right resources needed for the resolution of the current query. At this stage, agents will be launched (Step 5 on figure 6) to start a supervision process.

## VI. Medical Scenario

This section illustrates a rheumatologist processing an image through our distributed medical assistant. After connecting to the supervision server, the rheumatologist simply enters his query as keywords and uploads an image to analyze.

## A. *Preparation Phase*

On receiving a supervision query from a user, our system will translate it in a query language for the ontological architecture in order to determine the appropriate functionality (Step 2 on figure 6). Once determined, this functionality will be communicated to the supervision engine (Step 3 on figure 6) so it can decide the "good" knowledge files (Step 4 on figure 6) and thereafter, the right resources needed for the resolution of the current query. Let, for example, the selected knowledge files deal with a functionality having as a first composite operator, *OsteoMorph*. Then, the knowledge-based system launches this operator and starts a planning phase by decomposing this operator into its suboperators, as shown in figure 3: *Reading*, *Skeletonization* and *Analysis*. *Reading*, (1) in figure 3, is a primitive operator and does not need evaluation. The second sub-operator (*Skeletonization)* is a composite one, the system has to choose between two alternative sub-operators: *BinSkel* or Gray*Skel* (i.e. binary or gray-level skeletonization). Using expert choice rules given in the knowledge base, the engine selects one of them, say *BinSkel* (2). Since the knowledge base requires an evaluation of result for this operator, planning must be suspended and execution performed. At this stage, agents will be launched (Step 5 on figure 6) to start a supervision process.

## B. *Supervision Phase*

The execution starts with the Solver agent moving to the *Reading* program site and running it. Then the Solver goes to the *BinSkel* program site to execute it. Now, the evaluation phase can be performed. An Evaluator agent is created to execute evaluation rules. In case of automatic evaluation, it moves to the engine site to execute them. In case of user evaluation, it moves to the server site in order to ask the questions provided by these rules to the user. Then the Evaluator sends the assessment back to the engine. If the assessment is good, the planning phase continues with the *Analysis* composite operator. Otherwise, repair or adjustment rules are used to decide to modify the plan or to re-execute the same programs with different parameter values, resulting in a new plan. For instance, *BinSkel* assessment is manual.

The user is asked about the presence of undesirable segments. If this is the case an adjustment rule is fired, that increases the value of a pruning parameter, so that *BinSkel* may be executed again.

The same process runs up to the last program in the decomposition (5). The Solver agent executes this last operator and moves back to the server with the final result together with the plan established by the engine, for example, the following sequence of programs: *Reading*, *BinSkel*, *DimProj*, *Direction* and *AttributesDir*.

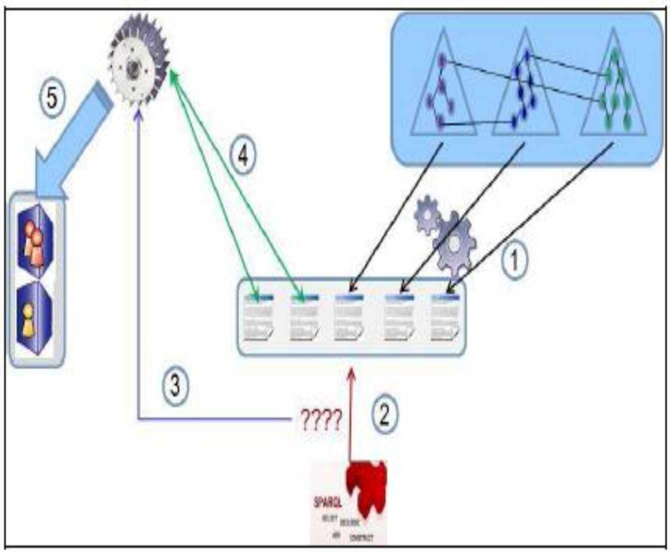


**Fig. 6. Ontologies integration and preparation phase.**

The system may also have other uses depending on the end-user type. For instance, image processing experts can use the supervision server to check their programs, observe their results, and compare them with other approaches. A collaborative construction of knowledge bases on the use of medical image processing programs is also possible.

## VII. Conclusion

Considering the early osteoporosis detection as a challenge for the medical community, we described in this paper, a distributed intelligent and interactive system relying on knowledge based techniques: (1) to assist physicians and image experts in validating "image protocols"; (2) to facilitate access to up to date image programs by rheumatologists who do not want to bother about medical image analysis details; (3) to allow physicians to submit an image, to obtain resulting medical parameters and also the corresponding execution plan (that is the effective "image protocol"). Therefore, the radiologists who are not specialists in image processing must be freed from the program details in order to focus on the interpretation of the results and their evaluation.

Distributing such systems is fundamental because physicians, image processing programs, images, inference engine, knowledge bases, etc. are generally located at different sites. Our distributed environment is based on a Web server, mobile agents for the communication inter-components and Semantic Web ontologies to facilitate physician access and knowledge exploration. Hence, the strength of the proposed system comes from the following points:

- Taking advantages from the technological advances in medical image processing.
- Facilitating the experiments by radiologist and physicians and allowing them to adjust and repair the values of the programs parameters.
- Making different experts work together.
- Consolidating the diagnosis of the disease.
- Taking into account the heterogeneity of data (images, knowledge, ontologies, programs, etc.).

Our tests dealt with small bone images (from 64 * 64 to 512 * 512). As prospects, we plan to improve the performance of our distributed and collaborative intelligent system when scaling with multiple queries and greater images, and this, in order to test its ability to maintain its functionalities and performance in an important demand.

## References

[1] M. Crubézy, F. Aubry, S. Moisan, V. Chameroy, M. Thonnat and R. Di Paola, "Managing complex processing of medical image sequences by program supervision techniques", In Proc. of SPIE Medical Imaging 1997, vol. 3035-85, pp. 614-625, Newport Beach, CA, February 1997.

[2] N. Khayati and W. Lejouad-Chaari, "A Distributed and Collaborative Intelligent System for Medical Diagnosis", Advances in Distributed Computing and Artificial Intelligence Journal **(**ISSN: 2255-2863**),** Volume 5, July 2013.

[3] N. Khayati and W. Lejouad-Chaari, "Agent and khnowledge models for a distributed imaging system", In proc. of the International Symposium on Distributed Computing and Artificial Intelligence, DCAI'2013 Salamanca, Spain. Advances in Intelligent and Soft Computing, Volume 217, May 2013, pp 235-242.

[4] N. Khayati, W. Lejouad-Chaari, S. Sevestre-Ghalila, "A Distributed Interactive Medical Diagnosis Support System", In Proceedings of the 2nd International Conference on Advanced Information and Telemedecine Technologies for Health (AITTH'2008), pp 59-63, Minsk, Belarus, October 2008.

[5] N. Khayati, W. Lejouad-Chaari, S. Sevestre-Ghalila, "A Distributed Image Processing Support System: Application to Medical Imaging", In Proceedings of the IEEE International Workshop on Imaging Systems and Techniques (IEEE-IST'2008), pp 261-264, Chania, Greece, September 2008.

[6] W. Lejouad-Chaari, S. Moisan, S. Sevestre-Ghalila, J.P. Rigaut, "Distributed Intelligent Medical Assistant for Osteoporosis Detection", In Proc. of the International Conference of IEEE Engineering in Medicine and Biology Society, Lyon – France, August 2007.

[7] C. Shekhar, S. Moisan and M. Thonnat, "Real-Time Perception Program supervision for Vehicle Driving Assistance", In Okyay Kaynak, Mehmed Ozkan, Nurdan Bekiroglu, and Ilker Tunay, editors, ICRAM'95 Intl. Conference on Recent Advances in Mechatronics, pp. 173–179, Istanbul.

[8] M. Thonnat, S. Moisan and M. Crubézy, "Experience in Integrating Image Processing Programs", Int. Conf. Vision Systems (ICVS'99). Las Palmas, Spain. January 1999. LNCS 1542, pp 200-215.

[9] M. Thonnat and S. Moisan, "What can Program Supervision do for Software Reuse?," IEEE Proc. Special Issue on Knowledge Modelling for Software Components Reuse, Vol. 5, No. 147, pp.179- 185, Oct. 2000.

[10] R. Vincent, M. Thonnat and J.C. Ossola, "Program supervision for automatic galaxy classification", In Proc. of the Intl. Conference on Imaging Science, Systems, and Technology, CISST'97, June 1997.

[11] S. Sevestre-Ghalila, A. Benazza, A. Ricordeau, N. Mellouli, C. Chappard et C.L. Benhamou, "Texture image analysis for osteoporosis detection with morphological tools". In P. Hellier C.Barillot, DR Haynor, editor, MICCAI 2004, volume 1, pages 87-94. LNCS Springer Verlag, September 2004.

[12] Moisan S, "Knowledge Representation for Program Reuse", ECAI'02, Lyon. France, Jul. 2002.